AMENDMENTS TO THE CLAIMS

(IN FORMAT COMPLIANT WITH THE REVISED 37 CFR 1.121)

Please add new claims 16-20.

A2

1. (CURRENTLY AMENDED) A method of recovering from ~~loading~~ invalid data ~~into~~ in a first register within a pipelined processor, the method comprising the steps of:

(A) setting a ~~register~~ first status for said first register to an invalid state in response to ~~loading invalid~~ a first data ~~into~~ in said first register being invalid; and

(B) stalling said processor in response to both (i) an instruction requiring said first data ~~buffered by said register~~ and (ii) said ~~register~~ first status being in said invalid state.

2. (CURRENTLY AMENDED) The method of claim 1, further comprising the step of:

~~(c)~~ setting said ~~register~~ first status ~~for said register~~ to said invalid state in response to a conditional store for said first data in said first register.

3. (CURRENTLY AMENDED) The method of claim 1, further comprising the step of:

~~(D)~~ setting said ~~register~~ first status ~~for said register~~ to said invalid state in response to said first register receiving a second data from ~~another~~ a second register having a second status in said invalid state.

4. (CURRENTLY AMENDED) The method of claim ~~1~~ 2, further comprising the steps of:

~~(E)~~ obtaining ~~valid~~ a second data ~~for said register~~ from a memory in response to ~~loading invalid data into said register~~ said conditional store; and

~~(F)~~ stalling said processor in response to completing said obtaining ~~valid~~ to enable said second data ~~for~~ to be written in said first register~~; and~~

~~(G) loading said register with valid data in response to stalling said processor in step (F)~~.

5. (CURRENTLY AMENDED) The method of claim 4, further comprising the ~~steps~~ step of:

~~(H)~~ stalling said processor prior to ~~writing new~~ obtaining said second data to prevent a third data from being written in said first register ~~having said invalid state while obtaining~~ before said ~~valid~~ second data is written in ~~for~~ said first register~~; and~~

~~(I) writing new data to said register in response to loading said register with valid data~~.

A2

6. (CURRENTLY AMENDED) The method of claim 1, further comprising the step of:

~~(J)~~ buffering said ~~register~~ first status as a plurality of bits to provide for multiple conditions that would ~~set said register status to~~ indicate said invalid state.

7. (CURRENTLY AMENDED) The method of claim 1, further comprising the steps of:

~~(C)~~ setting said ~~register~~ first status ~~for said register~~ to said invalid state in response to at least one of (i) a conditional store for said first data in said first register, (ii) receiving a second data from a second register having a second status in said invalid state and (iii) a miss for a data cache read of a third data to be written in said first register;

~~(D) setting said register status for said register to said invalid state in response to receiving data from another register having said invalid state;~~

~~(E)~~ obtaining ~~valid~~ said third data ~~for said register~~ from a memory in response to ~~loading invalid data into said register~~ at least one of said conditional store and said miss; and

~~(F)~~ stalling said processor in response to obtaining ~~valid~~ said third data to enable said third data to be written in ~~for~~ said first register~~; and~~

~~(G) loading said register with valid data in response to stalling said processor in step (F)~~.

A2

8. (CURRENTLY AMENDED) A pipelined processor comprising:

a first register configured to buffer (i) a first data and (ii) a ~~register~~ first status; and

logic configured to (i) set said ~~register~~ first status to an invalid state in response to ~~loading invalid~~ said first data ~~into said register,~~ being invalid and (ii) stall said processor in response to both (a) an instruction requiring said first data ~~buffered by said register~~ and (b) said ~~register~~ first status being in said invalid state.

9. (CURRENTLY AMENDED) The pipelined processor of claim 8 ~~further comprising:~~, wherein said logic ~~being~~ is further configured to ~~(iii)~~ set said ~~register~~ first status to said invalid state in response to a conditional store for said first data in said first register.

10. (CURRENTLY AMENDED) The pipelined processor of claim 8, further comprising~~.~~ a second ~~another~~ register~~, and~~, wherein said logic ~~being~~ is further configured to ~~(iv)~~ set said ~~register~~ first status to said invalid state in response to receiving a second data from said ~~another~~ second register having a second status in said invalid state.

A2

11. (CURRENTLY AMENDED) The pipelined processor of claim 8, further comprising~~.~~ a bus interface unit configured to obtain ~~valid~~ a second data ~~for said register~~ from a memory in response to ~~loading invalid~~ a miss for a read from a data cache of said second data to be written in ~~into~~ said first register~~, and~~, wherein said logic ~~being~~ is further configured to ~~(v)~~ (i) stall said processor in response to completing said obtaining ~~valid~~ of said second data ~~for said register,~~ and (ii) write said second data in ~~(vi) load~~ said first register ~~with valid data~~ in response to stalling said processor.

12. (CURRENTLY AMENDED) The pipelined processor of claim 11, ~~further comprising.~~ wherein said logic ~~being~~ is further configured to ~~(vii)~~ (i) stall said processor ~~prior to writing new data to said register~~ while said bus interface unit is said obtaining ~~valid~~ said second data ~~for said register,~~ in to prevent a third data from being written in said first register and ~~(viii)~~

(ii) write ~~new~~ said third data ~~to~~ in said first register in response to ~~loading said register with valid~~ writing said second data in said first register ~~obtained from said memory~~.

A2

13. (CURRENTLY AMENDED) The pipelined processor of claim 8, ~~further comprising:~~ wherein said ~~register~~ first status ~~being buffered as~~ comprises a plurality of bits to provide for multiple conditions that would ~~set said register status to~~ indicate said invalid state.

14. (CURRENTLY AMENDED) The pipelined processor of claim 8, further comprising:

a second ~~another~~ register; and

a bus interface unit configured to obtain ~~valid~~ a third data for said first register from a memory ~~in response to loading invalid data into said register; and~~, wherein said logic ~~being~~ is further configured to ~~(iii)~~ (i) set said ~~register~~ first status to said invalid state in response to a conditional store for said first data in said first register, ~~(iv)~~ (ii) set said ~~register~~ first status to said invalid state in response to receiving a second data from said ~~another~~ second register having a second status in said invalid state, ~~(v)~~ (iii) stall said processor in response to obtaining ~~valid~~ said third data for said first register, ~~(vi) load said register with valid~~ (iv) write said third

data in said first register in response to stalling said processor~~,~~ and ~~(vii)~~ (v) set said ~~register~~ first status to said invalid state in response to a cache load-miss for said first register.

A2

15. (CURRENTLY AMENDED) A pipelined processor comprising:

means for buffering a first data;

means for buffering a ~~register~~ first status;

means for setting said ~~register~~ first status to an invalid state in response to ~~loading invalid~~ said first data ~~into~~ in said means for buffering being invalid ~~data~~; and

means for stalling said processor in response to both (i) an instruction requiring said first data ~~buffered by said means for buffering data~~ and (ii) said ~~register~~ first status being in said invalid state.

16. (NEW) The method of to claim 1, further comprising the steps of:

setting said first status to said invalid state in response to a miss for a read from a data cache of a second data to be written in said first register.

17. (NEW) The method of to claim 16, further comprising the steps of:

obtaining said second data from a memory in response to said miss; and

stalling said processor in response to completing said obtaining to enable said second data to be written in said first register.

18. (NEW) The method of to claim 17, further comprising the step of:

stalling said processor prior to obtaining said second data to prevent a third data from being written in said first register before said second data is written in said first register.

19. (NEW) The pipeline processor of claim 13, wherein said logic comprises a first logic gate configured to combine said bits of said first status read from said first register.

20. (NEW) The pipeline processor of claim 8, wherein said logic comprises a first logic gate configured to combine a plurality of bits to form said first status written to said first register.